Computational and Structural Biotechnology Journal 20 (2022) 1487–1493

COMPUTATIONAL AND STRUCTURAL BIOTECHNOLOGY JOURNAL

journal homepage: www.elsevier.com/locate/csbj

SaAlign: Multiple DNA/RNA sequence alignment and phylogenetic tree construction tool for ultra-large datasets and ultra-long sequences based on suffix array

Ziyuan Wang [a,1], Junjie Tan [b,1], Yanling Long [c], Yijia Liu [a], Wenyan Lei [a], Jing Cai [d], Yi Yang [a], Zhibin Liu [a,*]

[a] Key Laboratory of Bio-Resource and Eco-Environment of Ministry of Education, College of Life Sciences, Sichuan University, Chengdu 610064, Sichuan, PR China
[b] Center for Clinical Molecular Medicine, National Clinical Research Center for Child Health and Disorders, Ministry of Education Key Laboratory of Child Development and Disorders, China International Science and Technology Cooperation Base of Child Development and Critical Disorders, Chongqing Key Laboratory of Pediatrics, Children's Hospital of Chongqing Medical University, Chongqing 400014, PR China
[c] College of Computer Science, Sichuan University, Chengdu 610064, Sichuan, PR China
[d] West China School of Pharmacy, Sichuan University, Chengdu 610041, Sichuan, PR China

ARTICLE INFO

Article history:
Received 5 October 2021
Received in revised form 9 March 2022
Accepted 19 March 2022
Available online 21 March 2022

Keywords:
Sequence analysis
Alignment
Phylogenetic tree
Suffix array

ABSTRACT

Multiple DNA/RNA sequence alignment is an important fundamental tool in bioinformatics, especially for phylogenetic tree construction. With DNA-sequencing improvements, the amount of bioinformatics data is constantly increasing, and various tools need to be iterated constantly. Mitochondrial genome analyses of multiple individuals and species require bioinformatics software; therefore, their performances need to be optimized. To improve the alignment of ultra-large datasets and ultra-long sequences, we optimized a dynamic programming algorithm using longest common substring methods. Ultra-large test DNA datasets, containing sequences of different lengths, some over 300 kb (kilobase), revealed that the Multiple DNA/RNA Sequence Alignment Tool Based on Suffix Tree (SaAlign) saved time and computational space. It outperformed the existing technical tools, including MAFFT and HAlign-II. For mitochondrial genome datasets having limited numbers of sequences, MAFFT performed the required tasks, but it could not handle ultra-large mitochondrial genome datasets for core dump error. We implement a multiple DNA/RNA sequence alignment tool based on Center Star strategy and use suffix array algorithm to optimize the spatial and time efficiency. Nowadays, whole-genome research and NGS technology are becoming more popular, and it is necessary to save computational resources for laboratories. That software is of great significance in these aspects, especially in the study of the whole-mitochondrial genome of plants.

1. Introduction

During biological information processing, similarities among sequences are used to construct phylogenetic trees for biological analyses [1]. In recent years, with the extreme increase in next-generation sequencing results, the data processing scale has grown from Mega Byte (MB) and Giga Byte (GB) to Terabyte (TB), PB, and even EB and ZB [2]. For instance, in metagenomics studies, millions of sequence reads are analyzed to determine the functional or taxonomic contents of microbial samples from the environment [3,4]. Thus, the performance levels of multi-sequence analysis and phylogenetic tree construction tools must improve [5].

Numerous sequence alignment analysis software packages are available online [6]. Existing state-of-the-art tools, such as MAFFT, PASTA (>200,000 sequences MSA), ProbPFP (PHMM model optimized by particle swarm optimization) and Minimap2 (developed for nanopore sequencing), allow sequence alignments to be run on a multi-thread workstation and for specific context. Additionally, numerous phylogenetic tree construction software programs are also being widely used in comparative genomics, cladistics, bioinformatics, and other fields [7–12]. MAFFT is a multiple sequence alignment program for Unix-like operating systems. It offers a range of multiple alignment methods, including L-INS-I and FFT-

Abbreviations: DP, Dynamic programming; LCS, Longest common subsequence; MSA, Multiple sequence alignment; SA, Suffix array.

* Corresponding author.
E-mail address: liuzhibin@scu.edu.cn (Z. Liu).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.csbj.2022.03.018

NS-2. Various iterations and optimizations to MAFFT have been made by contributors worldwide, and it is, at present, the most popular software for DNA and protein sequence alignments [7,8]. However, using MAFFT to align massive unrelated sequences is highly time consuming [13]. In addition, those unrelated sequences require more memory when using Dynamic programming algorithms if the spatial complexity of Needleman–Wunsch is $O(n^2)$ [13]. The user interface of MAFFT is still terminal based, allowing the user to manipulate and select the algorithms.

Some software packages use distributed computing frameworks, such as Hadoop and Spark, and they are gradually attracting more attention [14]. Sequential SparkBWA, in which the Spark cluster improves computing efficiency by using the BWA approach, enabled multi-node computing to improve computing performance [15].

For the global alignment of DNA sequences and the construction of phylogenetic trees, many faster approaches using Spark have been proposed, such as HAlign [16] and HAlign-II [17], that can efficiently build phylogenetic trees from large numbers of biological sequences and provide user-friendly web servers s through high-performance and distributed-computing infrastructures. However, these faster approaches also have some limitations. Using the Needleman-Wunsch algorithm to pairwise align DNA sequences illustrates high spatial complexity, which means only comparisons of satellite RNA, viral DNA, and genetic segments for eukaryotes can be performed, not whole-genome comparisons. Even if genome-wide comparisons could be made, they would require heavily loaded workstations and huge amounts of memory. Additionally, the code is implemented in JAVA, which is less efficient than software implemented in C language in terms of sequence comparison speed.

Owing to frequent exposures to various oxidation reactions, the variation rate of mitochondrial DNA is greater than that of nuclear DNA [18]. The resulting polymorphisms provide markers for comparisons of maternal DNA, and some polymorphic sites are related to the evolution of regional populations. Mitochondrial DNA is inherited completely from the maternal line, and there is no genetic recombination, so it can be used in the identification of maternal kinship [19]. As a result, a mitochondrial genome analysis of multiple individuals allows us to classify different species in a unique way; however, it requires more computing resources than using only a single gene. After investigating some algorithms for string processing, we optimized existing software using a distributed computing platform. We implemented our system, which supports multi-threading and multi-node computing, as well as allowing the adjustment of the allocated memory and the number of nodes as needed, to address ultra-large multiple biological sequence alignments and large-scale phylogenetic tree constructions. The system is based on the Apache Spark framework [20]. We adopt the optimized Needleman-Wunsch algorithm for pair-wise sequence alignments because it handles greater spatial complexity. Thus, the system can perform whole-mitochondrial genome sequence alignments and phylogenetic tree constructions.

This system provides an efficient and user-friendly tool with the following attributes:

A framework that runs Center Star Strategy on a Spark cluster, where the input data is streamed in parallel to the data nodes, like PCs and servers, executing the Needleman-Wunsch algorithm;
Presents phylogenetic trees in the popular Newick format [21];
The effects of the number of DNA sequences, sequence length, and data nodes on the acceleration ratio have been studied;
There is a graphical interface to increase user friendliness; and
5) The pair-wise sequence alignment algorithm and the phylogenetic tree construction algorithm are implemented by C language, which increases their efficiency levels.

2. Results

2.1. Data and measurements

Balibase is the golden benchmark for most MSAs [22]. This database, however, is relatively limited and is suitable for protein alignment only. Here, fungal ITS, complete viral, and mitochondrial genomes were used because there is no benchmark dataset for large-scale MSA issues. Because whole-genome sequence analyses require effort and computing resources, both gene fragments and complete genome datasets were included in this paper.

To test the performance of our program for large-scale data, we used three types of datasets consisting of 100 (1x), 1,000 (10x), and 10,000 (100x) sequences (No dataset contains two identical DNA sequences). In the fungal ITS genome 1x, 10x, and 100x datasets, there were 100, 1,000, and 10,000 fungal ITS sequences, respectively, with maximum and minimum lengths of 993 bp and 363 bp. This is a low similarity dataset.

To address DNA/RNA sequences with high similarity levels, we also tested our program on two Coronavirus datasets. The first contained 100 Coronavirus sequences, with an average length of 29,811.2 bp. The longest was 29,873 bp, while the shortest was 29,510 bp. The second dataset contained 1,000 Coronavirus sequences, with an average length of 29,800.8 bp. The longest was 29,901 bp, while the shortest was 29,407 bp. DNA datasets are shown in Table 1.

To address long DNA/RNA sequences of different lengths, we also tested our program on two mitochondrial genomes. The first was a 2.60 MB archive, while the second dataset was 29.18 MB. Detailed information regarding the experimental DNA datasets are shown in Table 1.

2.2. Comparison with state-of-the-art tools

Large-scale data cannot be handled by any of the available state-of-the-art MSA software tools. Therefore, only comparisons with MAFFT and HAlign-II were performed using clusters in which the node had 64 GB of memory, a 2.5 GHz 8 core CPU and a 64-bit Ubuntu Operating SSystem (Table S1).

The sum-of-pair score (SPS) counts what percentage pairs of residues are correctly aligned [23]. We use N denotes the number of sequences whose length is M. If in column i, sequence $\times$ and sequence y have residues aligned to the reference alignment's segment, then pair value P_{ixy} equals 2; if both alignments show a gap, then P_{ixy} equals 1, if mismatch P_{ixy} equals 0. The total score is normalized by the utmost possible score, in order that the range of possible values is from 0 to 1, with 1 indicating multiple alignments that's identical on the segments. The score S_i with the i th column and SPS are.

$$\begin{cases} S_i = \sum_{j=1}^{N} \sum_{k \neq j} P_{ijk} \\ SPS = \frac{\sum_{i=1}^{M} S_i}{\sum_{i=1}^{M_r} S_{ri}} \end{cases}$$

where M_r is the number of columns in the segments of the reference alignment and S_{ri} is the score S_i for the ith column in the reference alignment.

The increase in the number of sequences in the experimental dataset in this paper is different from previous experiments, and our datasets were achieved by adding non-repeating DNA sequences instead of simply repeating the original sequence set.

Table 1
Original dataset in the experiment.

Dataset	Max Length	Min Length	Average Length	Sequence Number	File Size
ITS sequences(1X)	738	459	572.83	100	65KB
ITS sequences(10X)	902	346	587.4	1000	659KB
ITS sequences(100X)	993	363	584.23	10,000	6.42MB
virus genome(1X)	29,873	29,510	29811.2	100	3MB
virus genome(10X)	29,901	29,407	29800.8	1000	26.75MB
Mitochondrion genome (small)	363,329	208,098	268591.1	10	2.60MB
Mitochondrion genome (large)	362,070	232,242	298624.64	100	29.18MB

Using the center star MSA algorithm, the center sequence will be longer, making the whole experiment more consistent with practical applications. The key goal of our work was to decrease the time required by an MSA and increase the ability to manage large volumes of data. Therefore, for large datasets, we concentrated on the running time of SaAlign, which was also performed on only one node.

In Table 2, the times required to analyze fungal ITS (a visualization of phylogenetic tree of ITS 1X shown in Fig. 1), complete viral, and mitochondrial genome datasets are shown. Different tools generated varied performance outcomes for different types of genomic datasets. Moreover, the sum-of-pairs value was chosen for measuring the alignment performance [23,24]. However, the SP value is not suited for massive multiple sequence alignment because the score may be very large, and for achieving better evaluation of ultra-large-scale alignments, we employ average SPS as the final metric of MSA experiments [17]. The accuracy is shown in Table 3, using the average sum-of-pair score (SPS) to measure. Finally, we give an example, which is a phylogenetic tree constructed by the FungiITS 1X dataset in Fig. 1 whose the lowest bootstrap values are greater than 77% [25,26].

To test more accurately, each tested software was used in multi-threading mode. SaAlign tools are multi-threaded tests in a single node, support multi-threaded calculations, and depend on the scheduling of the Spark platform. In addition to the SaAlign console version, there is also a web-based version, as well as a node management page through which parameters can be adjusted at any time.

MAFFT and HAlign-II perform well for the alignment of short DNA sequences (fungal ITS) when the number of sequences is not extremely large. Under our server parameters, MAFFT does not perform under abnormal conditions and results cannot be achieved through regular activities. The time consumed by HAlign-II was 49 min when handling the 100x fungal ITS dataset. Likewise, SaAlign took 57 min.

MAFFT performs well for the alignment of few Coronavirus sequences (100 Coronavirus sequences), taking 43.6 min to finish the alignment. HAlign was unable to run and stopped while handling coronavirus pairwise-alignment operations. When comparing a range of 100 coronavirus sequences, SaAlign was marginally slower than MAFFT (75.21 min vs 43.60 min, respectively), but SaAlign had advantages in the processing of large-scale coronavirus sequence datasets. It completed the alignment in 20.2 h. Neither HAlign nor MAFFT could adequately manage the alignment of 1,000 coronavirus sequences due to core dump/memory error.

For longer sequences, such as mitochondrion genome, SaAlign processed the two datasets provided here, while MAFFT operated when the number of sequences was small but was not allowed compare large numbers of sequences. Similar to the coronavirus genome dataset, HAlign-II was influenced by large sequence lengths during the pairwise alignment and was unable to process the complete mitochondrial genome.

2.3. Increasing the Spark platform's efficiency

In Fig. 2 and Fig. 3, the data showed that with an increase in nodes, the running time decreased, and memory efficiency significantly increased, indicating a linear growth in capacity and computing power along with nodes [27].

To determine the acceleration ratio of sequence set pairs in SaAlign with different sequence numbers on the Spark platform, we tested the fungal ITS datasets with one to five nodes in clusters. The dataset with the smallest number of sequences had a higher acceleration efficiency when the number of nodes was less than three. When the number of nodes exceeded four, the acceleration ratios of datasets having more sequences were greater. When the number of nodes was five, the acceleration ratio was greater than four for the 20x functional test dataset, and this was considerably greater than for the other two test datasets.

To determine the acceleration ratio of dataset pairs in SaAlign with different lengths of sequences on the Spark platform, we tested the 1x fungal ITS dataset, 1x Coronavirus dataset and the same 1x dataset with one to five nodes in the clusters. For the fungal ITS dataset (average length 572 bp), with the exception of the small acceleration ratio when the number of nodes was two, as the number of nodes increased, the inability to understand the acceleration resulted in an acceleration ratio of less than 1 owing to scheduling delays. For the other two datasets, their acceleration ratios increased along with the measured nodes.

3. Discussion

Here, we developed a MSA tool based on the center star strategy. To accelerate the MSA of incredibly close DNA sequences, we used a suffix array. It functioned easily but was unable to handle scalable data and genomes. It operated in parallel with Spark, which is an open framework for parallel programming. SaAlign has been shown by studies to manage >30,000 bp of sequencing data. The increasing numbers of sequences in the experimental datasets used here differs from previous experiments and were achieved by adding non-repeating DNA sequences instead of simply repeating the original sequence set. The key goal of our work was to increase the efficiency of the MSA and the ability to manage huge amounts of data. Therefore, for large data, we concentrated on decreasing the running time.

MAFFT and HAlign-II performed well for the alignment of short DNA sequences (fungal ITS) when the number of sequences was not extremely large. In addition, as the nodes increased more running time was saved. The processing of long sequences was more effective in SaAlign than HAlign-II, owing to the optimization of the algorithm based on the suffix array's LCS algorithm used in double-sequence comparisons. Additionally, using the Spark distributed framework helps in the management of huge quantities of files. Thus, SaAlign has more time to analyze very large datasets containing very long sequences. In addition, we used SPS to evaluate the accuracy. The result (Table 3) showed that SaAlign software

Table 2
Running time with genome MSA.

	ITS sequences (1X)	ITS sequences (10X)	ITS sequences (100X)	Virus genome (1X)	Virus genome (10X)	Mitochondrion genome(1X)	Mitochondrion genome(10X)
MAFFT	3.2 ± 0.1 s	14.32 ± 1.2 min	~	43.60 ± 2.8 min	~	7.3h	~
HAlign2.1	4.8 ± 0.3 s	8.41 ± 0.8 min	49.03 ± 2.2 min	~	~	~	~
SaAlign	23 ± 3.1 s	7.85 ± 1.3 min	57.25 ± 5.4 min	75.21 ± 4.9 min	11.6± 0.2h	1.81h	20.2h

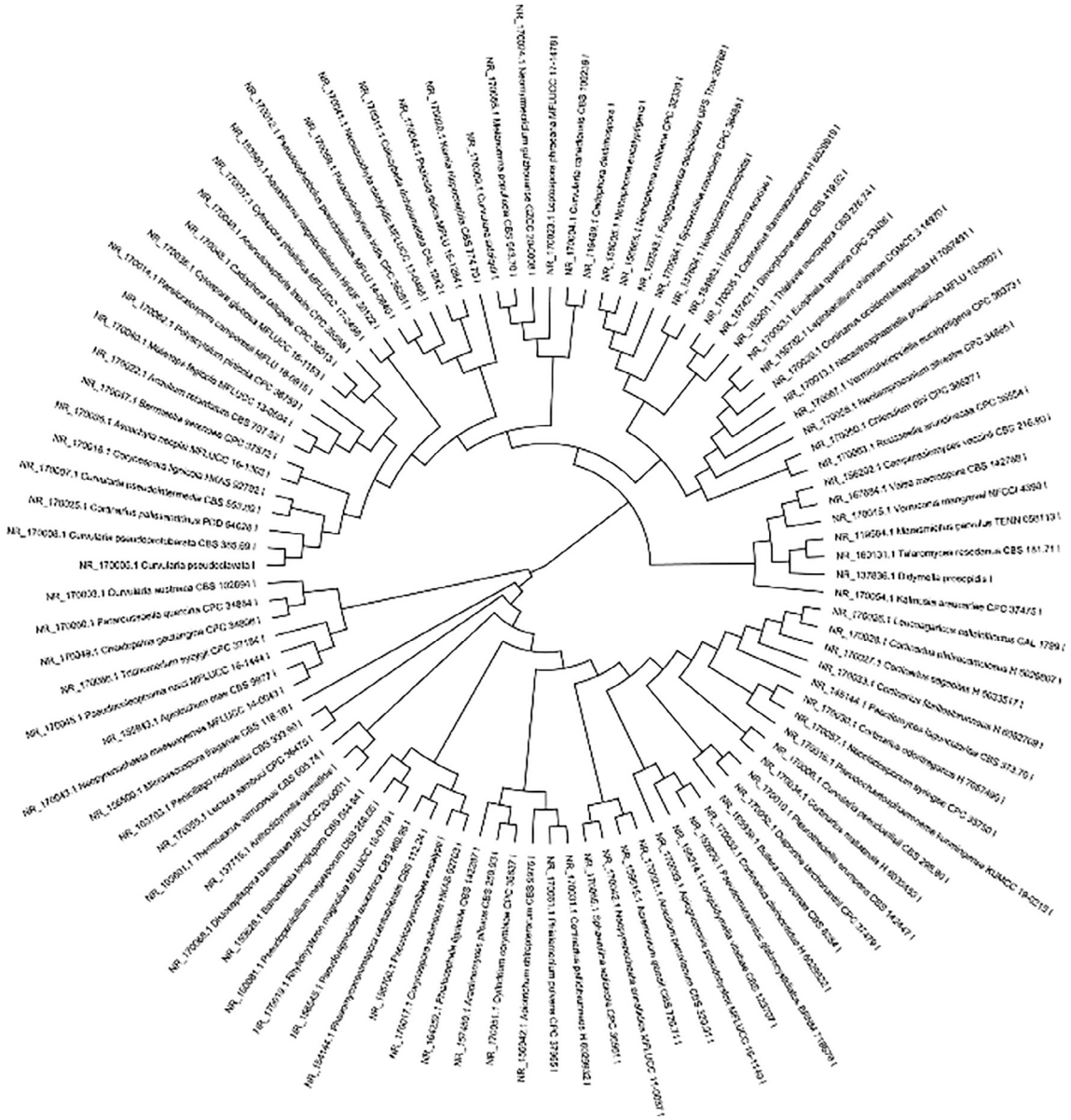

Fig. 1. Phylogenetic tree of fungiITS 1X dataset.

Table 3
Average SPS with genome MSA.

Avg SPS	ITS sequences (1X)	ITS sequences (10X)	ITS sequences (100X)	Virus genome (1X)	Virus genome (10X)	Mitochondrion genome (small)	Mitochondrion genome (large)
MAFFT	0.826	0.851	~	0.815	~	0.926	~
HAlign2.1	0.722	0.723	0.735	~	~	~	~
SaAlign	0.722	0.723	0.735	0.631	0.637	0.695	0.716

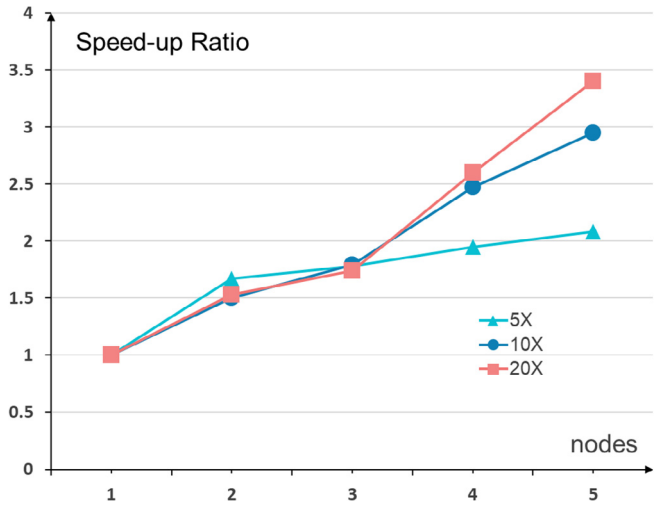

Fig. 2. Running time with increasing worker nodes using datasets consisting different number of sequences (5X, 10X and 20X denote dataset consisting 500, 1000 and 2000 sequences).

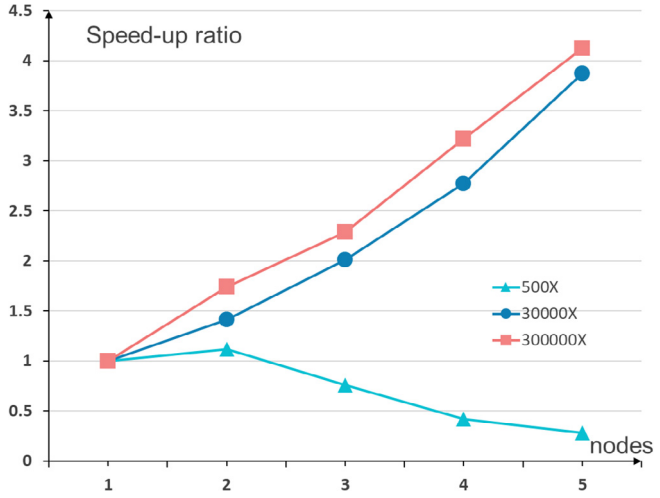

Fig. 3. Running time with increasing worker nodes using datasets consisting different length of sequences (500X, 30,000X and 300,000X denote dataset consisting sequences whose mean length are approximately 500, 30,000 and 300,000 bp).

had a good accuracy. Also, a bootstrap value of the phylogenetic trees (Fig. 1) higher than 70% also illustrated a good accuracy [25,26].

To investigate the influence of sequence number and length, we measured the SaAlign acceleration ratio. When the sequence dataset's average sequence length and nodes were short, the acceleration ratio was high. The acceleration ratio of the sequence dataset having a large number of sequences increased as the number of nodes expanded. The series length was longer when the estimation nodes were the same, and the acceleration ratio was high.

According to Amdal's law, if a portion of the program cannot be parallelized, acceleration will not be ideal [27]. The dataset with fewer sequences generated a shorter central sequence with the same average sequence length; therefore, the transmission overhead time was shorter. Additionally, the dataset with a reduced number of sequences had more benefits in acceleration than other datasets having two nodes. The estimated acceleration became comparatively significant as the number of nodes increases, and the degree of parallelization is high. The effect of overhead scheduling on the average running time decreased. Therefore, when the number of nodes was high, there was a greater acceleration ratio for sequence datasets with large numbers of sequences.

The time saved with long sequences was considerably greater for datasets having the same number of sequences than for central sequence transmission and scheduling. The propagation of the central sequence and scheduling time greatly impacted the acceleration ratio of the short sequence dataset. Additionally, the acceleration ratio decreased, even to less than 1, as the number of nodes increased. Clustering calculations had little impact on synchronization for sequences having an average length of 500 bp and was not even conducive to enhancing the calculation efficiency.

From the experiment with five nodes, it was predicted that, as the number of nodes increased, different sequence datasets would encounter a bottleneck of acceleration ratio growth. Therefore, we needed to conduct further performance-related experiments to optimize the selection of calculation nodes.

The tool was coded with C++ (DNA alignment algorithm) and python (PySpark). Spark 2.4.1 and Python 3.7 were required for use of the parallel tool.

Algorithms for phylogenetic tree construction often require MSA outcomes as input results, even though MSA algorithms use phylogenetic trees as guidance. Several free MSA phylogenetic tree construction algorithms have been suggested for huge unaligned DNA sequences, and they are often based on LCSs. The application of the suffix array to calculate the LCS was important for improving the performance of the sequence alignment. Protein sequences are usually short, and the identified LCSs do not improve the performance. However, for the analysis of metagenomic sequences, relatively long LCSs may be identified, and they would improve the computational performance. This method would be better suited for DNA sequences than protein sequences.

MSAs are important fundamental tools in bioinformatics, especially for phylogenetic tree construction. With improvements in DNA sequencing, the amount of bioinformatics data is constantly increasing, and various tools need to be iterated constantly. Other excellent methods have been developed to increase the computational time efficiency, even with a loss in precision loss, such as ClustalW-MPI [28], Hadoop-BAM [29], HAlign [16,17], and HPTree [30].The Smith-Waterman algorithm [31], suffix tree [32], and NJ (Neighbor-Joining) methods [33] are used to allow maximum use. MUSCLE [34], MAFFT [7,8], and Clustal-Omega [31] provide hardware resources and computational power using the Spark distributed and parallel computing paradigm. MUSCLE [34], MAFFT [7,8], and Clustal-Omega [31] have elevated accuracies for ultra-large genomes and RNA MSA studies. There were remarkable advantages in using distributed computing model methods, especially SaAlign, which provides the greatest memory performance. The experimental findings suggested that SaAlign performs better in terms of time efficiency, memory quality, and scalability with respect to ultra-large nucleotide.

4. Method and implementation

4.1. Overview of Apache Spark

Apache Hadoop and Apache Spark are well-known open-source frameworks in the field of distributed computing. Hadoop mainly includes the Hadoop-distributed file system for distributed storage and a parallel-programming model for large data sets. The Hadoop-distributed file system stores data on inexpensive machines, pro-

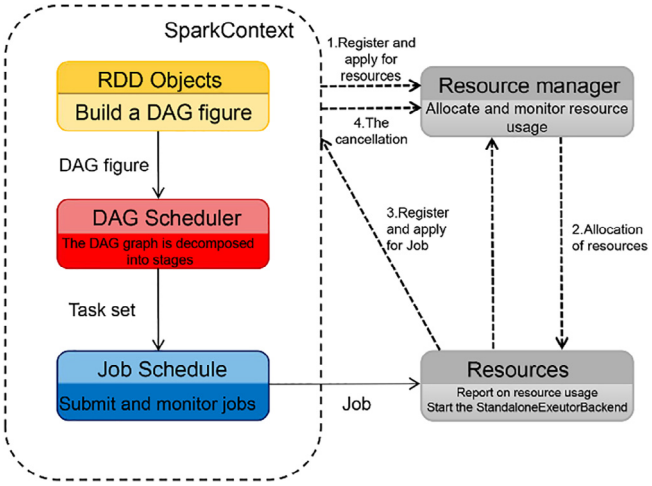

Fig. 4. A simple Spark workflow.

viding dependable fault-tolerant mechanisms and high-aggregate bandwidths across clusters, which are hidden from the users. The Spark workflow is presented in Fig. 4.

In the literature, the term "RDDs" [35] is used to refer to resilient distributed datasets, which is an abstract data structure of the Spark framework that stores data objects in a distributed data cluster. RDDs support an extensive variety of iterative algorithms, a highly efficient SQL engine, Shark, and a large-scale graph computing engine called GraphX [36]. With memory-based data storage and near-real-time processing power, Spark performs many times faster than other big data processing technologies. This might be because Spark stores intermediate results in memory rather than writing them to disk. To ensure a reliable tolerance for faults, RDDs are recomputed after data loss, such as the shutdown of individual machines. Using RDDs, Spark can achieve up to 100 times the theoretical speed of Hadoop using real data sets.

4.2. Overview of center star algorithm with Spark

The center star and progressive tree methods are two basic multiple DNA/RNA sequence alignment (MSA) strategies. The center star method runs faster, and it is, therefore, suitable for the MSA of similar DNA sequences [37]. The main approach underlying the center star method is to transform MSA into pairwise alignments based on a "center sequence". To accelerate the whole computing process, we used the Spark framework. The procedures are based on HAlign and HAlign-II; however, considering the limited memory of each computing node and the popularity of whole-genome alignments of bacteria having more than 420,000 bp of coding sequence, we adopted the optimized Needleman-Wunsch algorithm for analyzing pairwise sequences to reduce the cost of memory from O(mn) to O(min{m, n}). In addition, the optimization of the longest common substring (LCS) using a suffix array algorithm reduced the CPU time required.

Because the whole computing time is always considerable, to avoid a crash or system faults, the dataset is checked first. Characters except "ATCG" are viewed as illegal, and sequences containing these illegal characters are reported before the computing process begins. The procedures, based on the Spark distributed framework, to perform multiple sequences alignments are shown in Fig. 1 Supplementary. The neighbor-joining method is a widely used method for constructing phylogenetic trees [33]. Neighbor-joining was used to construct a phylogenetic tree to verify the contribution of our MSA algorithm toward evolutionary analysis.

4.3. Overview of the suffix array

To identify the LCS of two strings, we used a suffix array [38,39]. First, we concatenated two strings with a character in the middle that does not appear in either string. The purpose was to separate the two strings when later identifying the prefix. Then, we determined the suffix and the height arrays of the new concatenated strings. The height array represents the longest common prefix of the two suffixes adjacent to the rank. We identified the maximum value in the height array and then determined whether the prefix meets the requirements (we cannot judge whether the longest prefix is derived from two strings or from one string). Using the suffix array, we screened whether the selected longest prefix originated from two strings. Finally, we ascertained the LCS from the two strings.

4.4. Overview of the Needleman-Wunsch algorithm based on suffix array optimization

The Needleman-Wunsch algorithm is a classic and widely used algorithm to solve double-sequence alignment problems through the use of dynamic programming. However, the space and time complexity of common dynamic programming algorithms are not ideal when facing long-sequence pairwise problems. To reduce the memory load, the divide and conquer concept was used through an optimized Needleman-Wunsch algorithm.

If the aligned sequences S1 and S2 are long, then they were aligned recursively. Otherwise, they were pairwise aligned using the Needleman-Wunsch algorithm and the recursiveness was eliminated. First, the suffix array was used to determine the LCS of S1 and S2. After the LCS sequence was found, S1 was divided into Pre-S1 and S1-Post, and S2 was divided into Pre-S2 and Post-S2. The above pairwise alignment process was repeated for two groups of sequences (Pre-S1 and Pre-S2) and (S1-Post and Post-S2), as well as splice Pre-S1, LCS, and S1-Post to obtain S1. The same procedure was used to obtain S2. The pairwise-alignment model is shown in Fig. 5 and the concrete pseudocodes are presented in Supplementary method.

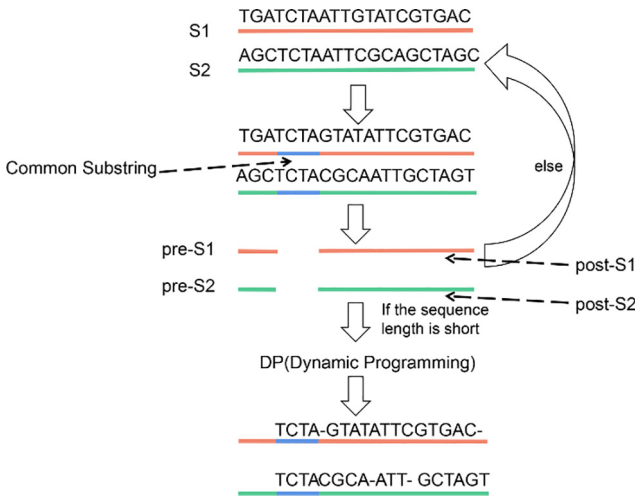

Fig. 5. Alignment procedure of Needleman-Wunsch algorithm optimization by longest common substrings.

CRediT authorship contribution statement

Ziyuan Wang: Conceptualization, Methodology, Software, Validation, Writing – original draft. **Junjie Tan:** Data curation, Writing – review & editing. **Yanling Long:** Visualization. **Yijia Liu:** Visualization. **Wenyan Lei:** Software, Validation. **Jing Cai:** Supervision. **Yi Yang:** Supervision. **Zhibin Liu:** Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by funding from Sichuan Science and Technology Program (No. 2021YFN0119), National Natural Science Foundation of China (Grant No. 31870240).

Data availability

Software Source Code: https://github.com/YangyiLab/SaAlign.
DNA Sequences: https://github.com/YangyiLab/SaAlign.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.csbj.2022.03.018.

References

[1] Hong Y, Guo M, Wang J. ENJ algorithm can construct triple phylogenetic trees. Mol Ther Nucleic Acids 2021;23:286–93. https://doi.org/10.1016/j.omtn.2020.11.004.
[2] Kolomvatsos K. A distributed, proactive intelligent scheme for securing quality in large scale data processing. Computing 2019;101:1687–710. https://doi.org/10.1007/s00607-018-0683-9.
[3] Wooley JC, Godzik A, Friedberg I. A primer on metagenomics. PLoS Comput Biol 2010;6:. https://doi.org/10.1371/journal.pcbi.1000667e1000667.
[4] Wooley JC, Ye Y. Metagenomics: facts and artifacts, and computational challenges. J Comp Sci Technol 2010;25:71–81. https://doi.org/10.1007/s11390-010-9306-4.
[5] Godini R, Fallahi H. A brief overview of the concepts, methods and computational tools used in phylogenetic tree construction and gene prediction. Meta Gene 2019;21:. https://doi.org/10.1016/j.mgene.2019.100586100586.
[6] Smith DR. Buying in to bioinformatics: An introduction to commercial sequence analysis software. Briefings Bioinf 2014;16:700–9. https://doi.org/10.1093/bib/bbu030.
[7] Nakamura T, Yamada KD, Tomii K, Katoh K. Parallelization of MAFFT for large-scale multiple sequence alignments. Bioinformatics 2018;34:2490–2. https://doi.org/10.1093/bioinformatics/bty121.
[8] Katoh K, Misawa K, Kuma KI, Miyata T. MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res 2002;30:3059–66. https://doi.org/10.1093/nar/gkf436.
[9] Pal J, Ghosh S, Maji B, Bhattacharya DK. Use of FFT in protein sequence comparison under their binary representations. Comput Mol Biosci 2016;06:33–40. https://doi.org/10.4236/cmb.2016.62003.
[10] Mirarab S, Nguyen N, Guo S, Wang L-S, Kim J, Warnow T. PASTA: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences. J Comput Biol 2014;22:377–86. https://doi.org/10.1089/cmb.2014.0156.
[11] Zhan Q, Wang N, Jin S, Tan R, Jiang Q, Wang Y. ProbPFP: a multiple sequence alignment algorithm combining hidden Markov model optimized by particle swarm optimization with partition function. BMC Bioinf 2019;20:1–10. https://doi.org/10.1186/s12859-019-3132-7.
[12] Li H. Minimap2: Pairwise alignment for nucleotide sequences. Bioinformatics 2018;34:3094–100. https://doi.org/10.1093/bioinformatics/bty191.
[13] Jiang X, Fu X, Dong G, Li H. Research on Pairwise Sequence Alignment Needleman-Wunsch Algorithm 2017;141:1041–6. 10.2991/icmmcce-17.2017.187.
[14] Lu HC, Hwang FJ, Huang YH. Parallel and distributed architecture of genetic algorithm on Apache Hadoop and Spark. Appl Soft Comp J 2020;95:. https://doi.org/10.1016/j.asoc.2020.106497106497.
[15] Abuín JM, Pichel JC, Pena TF, Amigo J. SparkBWA: speeding up the alignment of high-throughput DNA sequencing data. PLoS ONE 2016;11:. https://doi.org/10.1371/journal.pone.0155461e0155461.
[16] Zou Q, Hu Q, Guo M, Wang G. HAlign: Fast multiple similar DNA/RNA sequence alignment based on the centre star strategy. Bioinformatics 2015;31:2475–81. https://doi.org/10.1093/bioinformatics/btv177.
[17] Wan S, Zou Q. HAlign-II: Efficient ultra-large multiple sequence alignment and phylogenetic tree reconstruction with distributed and parallel computing. Algorithms Mol Biol 2017;12:1–10. https://doi.org/10.1186/s13015-017-0116-x.
[18] Song Q, Du Y, Liu Y, Zhang G, Wang Y. Complete mitochondrial genome of Aspergillus japonicus from the built environment and its phylogenetic analysis. Mitochondrial DNA Part B 2020;5:1445–6. https://doi.org/10.1080/23802359.2020.1735972.
[19] Merheb M, Matar R, Hodeify R, Siddiqui SS, Vazhappilly CG, Marton J, et al. Mitochondrial DNA, a powerful tool to decipher ancient human civilization from domestication to music, and to uncover historical murder cases. Cells 2019;8. https://doi.org/10.3390/cells8050433.
[20] Abuín JM, Lopes N, Ferreira L, Pena TF, Schmidt B. Big Data in metagenomics: Apache Spark vs MPI. PLoS ONE 2020;15:e0239741.
[21] Junier T, Zdobnov EM. The Newick utilities: high-throughput phylogenetic tree processing in the Unix shell. Bioinformatics 2010;26:1669–70. https://doi.org/10.1093/bioinformatics/btq243.
[22] Thompson JD, Plewniak F, Poch O. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. Bioinformatics 1999;15:87–8. https://doi.org/10.1093/bioinformatics/15.1.87.
[23] Carrillo H, Lipman D. The multiple sequence alignment problem in biology. SIAM J Appl Math 1988;48:1073–82. https://doi.org/10.1137/0148063.
[24] Darling AE, Mau B, Perna NT. progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. PLOS ONE 2010;5:e11147.
[25] Efron B, Halloran E, Holmes S. Bootstrap confidence levels for phylogenetic trees. PNAS 1996;93:13429–34. https://doi.org/10.1073/pnas.93.23.13429.
[26] Soltis PS, Soltis DE. Applying the bootstrap in phylogeny reconstruction. Statistical Sci 2003;18:256–67. https://doi.org/10.1214/ss/1063994980.
[27] Hill MD, Marty MR. Amdahl's law in the multicore era. Computer 2008;41:33–8. https://doi.org/10.1109/MC.2008.209.
[28] Li K-B. ClustalW-MPI: ClustalW analysis using distributed and parallel computing. Bioinformatics 2003;19:1585–6. https://doi.org/10.1093/bioinformatics/btg192.
[29] Niemenmaa M, Kallio A, Schumacher A, Klemelä P, Korpelainen E, Heljanko K. Hadoop-BAM: directly manipulating next generation sequencing data in the cloud. Bioinformatics 2012;28:876–7. https://doi.org/10.1093/bioinformatics/bts054.
[30] Zou Q, Wan S, Zeng X. HPTree: Reconstructing phylogenetic trees for ultra-large unaligned DNA sequences via NJ model and Hadoop. 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2016, p. 53–8. 10.1109/BIBM.2016.7822492.
[31] Pearson WR. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. Genomics 1991;11:635–50. https://doi.org/10.1016/0888-7543(91)90071-L.
[32] Su W, Liao X, Lu Y, Zou Q, Peng S. Multiple sequence alignment based on a suffix tree and center-star strategy: a linear method for multiple nucleotide sequence alignment on spark parallel framework. J Comput Biol 2017;24:1230–42. https://doi.org/10.1089/cmb.2017.0040.
[33] Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 1987;4:406–25. https://doi.org/10.1093/oxfordjournals.molbev.a040454.
[34] Edgar RC. MUSCLE: A multiple sequence alignment method with reduced time and space complexity. BMC Bioinf 2004;5:1–19. https://doi.org/10.1186/1471-2105-5-113.
[35] Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of NSDI 2012: 9th USENIX Symposium on Networked Systems Design and Implementation. p. 15–28.
[36] Xin RS, Gonzalez JE, Franklin MJ, Stoica I. GraphX: A resilient distributed graph system on spark. 1st International Workshop on Graph Data Management Experiences and Systems, GRADES 2013 – Co-Located with SIGMOD/PODS 2013 2013. 10.1145/2484425.2484427.
[37] Sun Y, Zhang Z, Wang J. A novel algorithm for DNA multiple sequence alignment based on the sliding window and the keyword tree. Int J Biosci, Biochem Bioinform 2013;3:271–5. https://doi.org/10.7763/ijbbb.2013.v3.211.
[38] Na JC, Park H, Lee S, Hong M, Lecroq T, Mouchard L, et al. Suffix array of alignment: A practical index for similar data. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2013;8214 LNCS:243–54. 10.1007/978-3-319-02432-5_27.
[39] Bingmann T. Scalable string and suffix sorting. Algorith. Techn. Tools 2018:1–396.